UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Dejan Vrhovnik

Razvoj orodja za centralni nadzor posodobitev v platformi WordPress

DIPLOMSKO DELO

VISOKOŠOLSKI STROKOVNI ŠTUDIJSKI PROGRAM PRVE STOPNJE RAČUNALNIŠTVO IN INFORMATIKA

Ljubljana 2014

UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Dejan Vrhovnik

Razvoj orodja za centralni nadzor posodobitev v platformi WordPress

DIPLOMSKO DELO

VISOKOŠOLSKI STROKOVNI ŠTUDIJSKI PROGRAM PRVE STOPNJE RAČUNALNIŠTVO IN INFORMATIKA

MENTOR: viš. pred. dr. Igor Rožanc

Ljubljana 2014

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika dela:

V diplomski nalogi predstavite razvoj orodja za centralni nadzor nad nameščanjem posodobitev za večje število neodvisnih spletnih strani v platformi WordPress. Rešitev naj bo izvedena z vtičnikom za centralno spletno stran in vtičnikom za nadzorovane spletne strani, omogoča pa naj prikaz stanja namestitev vseh strani ter nameščanje posodobitev na avtomatski ali ročni način. Nalogo zaključite s prikazom učinkovitosti rešitve.

MENTOR: viš. pred. dr. Igor Rožanc

IZJAVA O AVTORSTVU DIPLOMSKEGA DELA

Spodaj podpisani Dejan Vrhovnik, z vpisno številko **63990379**, sem avtor diplomskega dela z naslovom:

Razvoj orodja za centralni nadzor posodobitev v platformi WordPress

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom viš. pred. dr. Igorja Rožanca,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki "Dela FRI".

V Ljubljani, dne 15. april 2014 Podpis avtorja:

KAZALO

KAZALO SLIK

SEZNAM UPORABLJENIH KRATIC

- **API** (angl. *Application Programming Interface*) – programski vmesnik aplikacije,
- **CMS** (ang. *Content Management System*) – sistem za upravljanje vsebin,
- **CSS** (angl. *Cascading Style Sheets*) - stilske predloge, ki določajo izgled spletnih strani,
- **HTTP** (angl. *Hypertext Transfer Protocol*) - internetni komunikacijski protokol, namenjen za izmenjavo besedila in grafičnih, zvočnih ter drugih večpredstavnostnih vsebin na spletu,
- **JavaScript** - skriptni programski jezik za razvoj interaktivnih spletnih strani,
- **jQuery** - knjižnica za skriptni jezik JavaScript, ki prinaša zbirko funkcij za hitrejši razvoj spletnih aplikacij,
- **JSON** (angl. *JavaScript Object Notation*) - preprost format za izmenjavo podatkov,
- **MO** (angl. *Machine Object*) – optimizirana binarna predloga, katera vsebuje končne prevode teksta vtičnikov in tem,
- **MySQL** (angl. *My Structured Query Language*) - odprtokodni sistem za upravljanje zbirk podatkov,
- **PHP** (angl. *PHP Hypertext Preprocessor*) - skriptni programski jezik, namenjen izdelavi dinamičnih spletnih strani,
- **PO** (angl. *Portable Object*) – prenosna predloga za jezikovne prevode vtičnikov in tem,
- **POT** (angl. *Portable Object Template*) – urejena predloga za jezikovne prevode,
- **URL** (angl. *Uniform Resource Locator*) - predstavlja enoličen naslov vira,
- **UTF-8** (angl. *Universal Character Set Transformation Format 8-bit*) – oktetna predstvitev nabora znakov.

POVZETEK

Cilj diplomske naloge je razvoj orodja za centralni nadzor posodobitev v platformi WordPress, kjer je možen pregled in izvedba posodobitev stanja večjega števila med seboj neodvisnih platform WordPress.

V uvodu je predstavljena platforma WordPress ter problematika nadzora posodobitev pri večjemu številu strani. Prikazana je izdelava vtičnika za platformo, s katerim se razširi funkcionalnost same platforme. Izdelana rešitev je sestavljena iz dveh sklopov: iz vtičnika za centralno spletno stran in iz vtičnika za spletne strani, nad katerimi se izvaja nadzor posodobitev. Opisana je komunikacija odjemalec – strežnik, ki poteka med centralno stranjo in nadzorovanimi stranmi. Omogoča zajem podatkov stanja platform nadzorovanih spletnih strani, ki se prikažejo na centralni spletni strani. Sledi opis funkcionalnosti in delovanje vtičnika za centralno stran ter opis funkcionalnosti in delovanja vtičnika za nadzorovane strani. Na koncu je podana ocena uporabnosti izdelane rešitve ter predlogi za nadaljnje delo.

Ključne besede: WordPress, CMS, platforma, vtičnik, oddaljeni dostop, strežnik-odjemalec

ABSTRACT

The objective of the diploma thesis is the development of a tool to enable the central management of updates for the WordPress platform by adding the control over distribution of changes to a multitude of independent WordPress platforms.

The introduction presents the WordPress platform and the problems concerning the control of updates to a large number of webpages. It shows the development of a plug-in solution, which extends the functionality of the platform itself. The solution consists of two components: a plug-in for the central website and a plug-in for websites for which control has to be provided. First, the client – server communication is explained, which runs between the central and the supervised websites. It enables the collection of the state data for the controlled platforms displayed on the central website. Next, the explanation of the functionality and operation of the plug-in for the central website and controlled websites is provided as well. Finally, the thesis gives an assessment of the usefulness of the developed solution, and propose suggestions for further work.

Keywords: WordPress, CMS, platform, plug-in, remote access, client-server

1. UVOD

Spletna stran je danes postala pomemben medij za komunikacijo s svetom, preko nje želimo predstaviti sebe, svoje delo, podjetje ali pa uporabo spletne strani za trženje preko spleta. Razvoj orodij za izdelavo spletnih strani je zelo olajšal in pohitril njihovo izdelavo. Na tržišču je veliko izpeljank sistemov za upravljane vsebin (v nadaljevanju platforma CMS), ki temeljijo na osnovi PHP in MySQL tehnologije. Platforme so v večini odprtokodne in brezplačne.

Priljubljene platforme CMS:

- WordPress [1],
- Joomla [2],
- Typo3 [3],
- Drupal [4],
- Prestashop [5].

WordPress je odprtokodna platforma, ki je zgrajena na osnovi PHP in MySQL tehnologije in je brezplačen [1]. Njegovi začetki segajo v leto 2003. Razvil se je iz platforme b2 oz. cafelog [6]. Sprva je bil namenjen predvsem za pisanje in objavljanje blogov, vendar se je zaradi hitrega razvoja začel uporabljati tudi v druge namene. Danes se WordPress uporablja za predstavitvene strani, spletne trgovine, itd. Zaradi enostavne namestitve, preprostega administriranja in dobre podpore, je primeren tako za zahtevne in vešče kot tudi za preproste uporabnike, ki se želijo s pomočjo WordPress platforme predstaviti na spletu.

WordPress deluje na podlagi sistema vtičnikov in sistema spletnih predlog, kar zagotavlja hitro in enostavno prilagoditev funkcionalnosti in podobe spletne strani. Za vse skupaj skrbi jedro WordPressa.

V jedru so zajete funkcije, ki jih platforma uporablja za svoje delovanje. Velik nabor le teh omogoča prilagodljivost in stabilnost delovanja z bazo in datotečnim sistemom. Uporabniku omogoča nadzor nad sistemom, vtičniki, temami in jezikovno podporo.

Vtičniki omogočajo razširitev funkcionalnosti osnovne platforme. Glede na specifične potrebe uporabnika, se z aktivacijo vtičnika ponudijo dodatne funkcije. Na ta način vtičnik lahko vpliva na jedro sistema ali pa na delovanje drugega vtičnika in tako razširi funkcionalnosti. Mesto namestitve vtičnika je standardizirano in določeno s strani platforme.

Na vizualno podobo spletne strani vpliva nameščena tema. Omogoča nam spremembo podobe spletne strani le z namestitvijo in aktivacijo druge teme. Vsebina strani se pri tem ne spreminja.

Za varnost in zanesljivost spletne strani in s tem tudi WordPressa je potrebno poskrbeti tako, da je vedno nameščena zadnja različica jedra. Prav tako morajo biti nameščene zadnje

različice vtičnikov in tem. V nasprotnem primeru se lahko zgodi, da se sistem okuži z zlonamerno kodo in v najslabšem primeru preneha delovati. Ker je WordPress odprtokodni sistem, se tudi možnost vdora v sistem poveča, če posodobitve niso nameščene.

Sistem nas sam opozori, kdaj je na voljo nova posodobitev jedra, vtičnika, teme ali jezikovna posodobitev. Do prihoda različice 3.7 je bila na voljo le ročna posodobitev, sedaj pa je na voljo tudi samodejna posodobitev. Za aktivacijo samodejnih posodobitev jedra, vtičnikov in tem je potrebna še ročna sprememba nastavitev v PHP datoteki. Pri vsaki posodobitvi obstaja tveganje, da se sistem po posodobitvi poruši ali ne deluje več optimalno. Za občutljive vtičnike ali teme si zato želimo ročno posodabljanje, ker je to bolj pod nadzorom.

Imeti nadzor nad posodobitvami je ključnega pomena za optimalno delovanje WordPressa. Pri večjem številu spletnih strani narejenih v WordPressu, je potrebno vstopiti v vsako posamezno administracijo in izvesti posodobitve, če so na voljo. Rešitev tega problema je centralna administracija, v kateri je seznam spletnih strani in njihove posodobitve. Posodobitve v praksi terjajo kar nekaj pozornosti in truda. Več strani kot jih imamo v lasti, več dela je s posodobitvami.

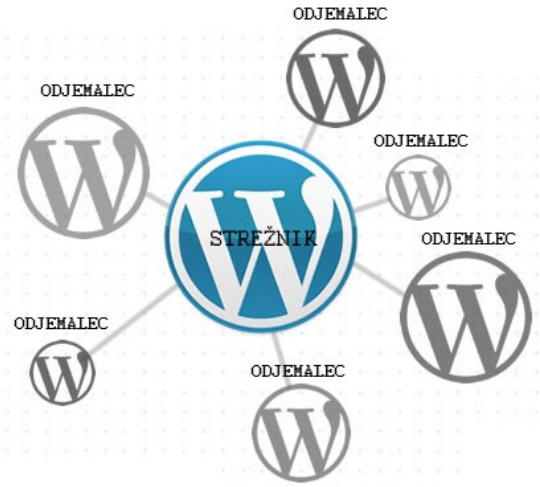
Na razpolago je kar nekaj rešitev za urejanje večjega števila WordPress strani na istem mestu. Primeri nekaterih rešitev:

- WP Remote [7],
- InfiniteWP [8],
- Manage WP [9],
- iControlWP [10] ali
- CMS Commander [11].

Pri pregledu obstoječih rešitev smo opazili, da nam omogočajo le namestitev vtičnika odjemalca. Za dostop do strežnika se moramo tako na njihovi strani registrirati in prijaviti v strežniški vmesnik. Različni ponudniki ponujajo različne možnosti v samem vmesniku, nekatere od njih so plačljive. Zaradi varnosti spletnih strani ne želimo, da bi imele druge osebe dostop do naših spletnih strani oziroma podatkov.

2. RAZVOJ REŠITVE

Z razvojem rešitve želimo poenostaviti in pohitriti pregled nad posodobitvami ter njihovo izvedbo. Zato želimo celovito rešitev, pri kateri imamo nadzor na celotnim sistemom, tako nad centralno stranjo, kot nad stranmi, pri katerih želimo nadzirati posodobitve. Shemo naše rešitve prikazuje slika 2.1. Centralni nadzor nad posodobitvami narekuje implementacijo dveh vtičnikov za platformo Wordpress. Za centralno stran potrebujemo *vtičnik strežnik* in za nadzorovane spletne strani postavljene v WordPressu *vtičnik odjemalec*, ki medsebojno komunicirata in si izmenjujeta podatke. Z implementacijo rešitve kot vtičnik, dosežemo preprosto nadgradnjo obstoječe spletne strani v centralno stran oziroma nadgradnjo obstoječe spletne strani v odjemalca.



Slika 2.1: Shema nadzora posodobitev

Vtičnik za centralno stran (v nadaljevanju *vtičnik strežnik*) služi za komunikacijo s spletnimi stranmi oziroma s svojimi odjemalci. Prikazuje seznam odjemalcev, nad katerimi se izvaja nadzor posodobitev. Za vsakega odjemalca nudi prikaz:

- seznama nameščenih vtičnikov,
- seznama nameščenih tem,
- seznama posodobitev jedra,
- seznama posodobitev vtičnikov in
- seznama posodobitev tem

Vtičnik odjemalec je nameščen na vsako spletno stran narejeno v WordPressu, nad katero se izvaja nadzor posodobitev. Komunicira s strežnikom pri registraciji odjemalca v sistem, izvaja njegove zahteve in odgovarja nanje.

2.1 RAZVOJNO OKOLJE

Razvoj rešitve je potekal s pomočjo urejevalnika teksta Notepad++ [12]. To je preprost, vendar napreden urejevalnik teksta, ki je prilagojen za različne programske jezike. Lahko bi

uporabili tudi druga bolj ali manj napredna razvojna okolja, vendar je za potrebe izdelave vtičnika zadoščal omenjeni urejevalnik teksta. Pri razvoju so uporabljene različne tehnologije. Osnova vtičnika je narejena v PHP jeziku, Uporablja vgrajene funkcije in metode platforme, katere so potrebne za izdelavo vtičnika, komunikacijo med strežnikom in odjemalci, zajemu podatkov odjemalcev. Za grafično podobo so uporabljena slogovna polja (v nadaljevanju slog CSS), knjižnica JQuery [13]. Knjižnica JQuery vsebuje že vnaprej napisane funkcije za skriptni jezik JavaScript in omogoča hitrejši razvoj programske opreme.

Vtičnika sta narejena in preizkušena v WordPressu različic 3.7.1, 3.8 in 3.8.1 v angleškem in slovenskem jeziku [14]. Testno okolje WordPress je nameščeno na Linux sistemu slovenskega ponudnika gostovanja.

2.2 IZDELAVA VTIČNIKA

Pri izdelavi vtičnikov se je potrebno držati pravil, ki jih narekuje *kodeks za WordPress* [15]. Ta pravila opredeljujejo standarde, po katerih delujejo jedro platforme, vtičniki in teme. Vtičniki se v WordPressu nahajajo v mapi `wp-content/plugins` in so razvrščeni v svojih podmapah. Če se pravil ne upošteva, tak vtičnik ne deluje pravilno in lahko onemogoča pravilno delovanje drugih vtičnikov ali celo ogrozi stabilnost celotne platforme.

Programska koda v PHP datoteki mora biti pisana v kodnem naboru UTF-8 in vsebovati informacijsko glavo vtičnika (angl. *plugin information header*) [16]. Primer informacijske glave vtičnika prikazuje koda 2.1. WordPress tako prepozna obstoj vtičnika in ga doda na zaslon upravljanja z vtičnikom (angl. *plugin management screen*), kjer vtičnik aktiviramo.

```
/**
* Plugin Name: Name Of The Plugin
* Plugin URI: http://URI_Of_Page_Describing_Plugin_and_Updates
* Description: A brief description of the Plugin.
* Version: The Plugin's Version Number, e.g.: 1.0
* Author: Name Of The Plugin Author
* Author URI: http://URI_Of_The_Plugin_Author
* License: A "Slug" license name e.g. GPL2
*/
```

Koda 2.1:Primer informacijske glave vtičnika

Za delo z vtičnikom ima WordPress pester nabor funkcij, poznane kot kljuke (angl. *hooks*). Poznamo dve vrsti kljuk: akcije (angl. *actions*) in filtre (angl. *filters*).

Akcije se prožijo ob posebnih dogodkih v WordPressu, kot je objava prispevka, sprememba teme, pošiljanje elektronskega sporočila, itd. [16]. Akcija je po meri narejena funkcija PHP, definirana v vtičniku in ujeta ob proženju dogodka. Funkcija prejme vsaj en parameter in vrne rezultat na mestu klica.

Pri poimenovanju funkcij je potrebno paziti, da ne pride do spora s funkcijo z istim imenom, saj PHP ne dopušča več funkcij z istim imenom. Spor se lahko prepreči na dva načina. Funkcije se poimenuje z edinstveno predpono oziroma se za to uporabi razred (angl.

class). Definirano funkcijo je na koncu potrebno ujeti oziroma registrirati v WordPressu, s pomočjo metode `add_action()`. Ujetje funkcije prikazuje koda 2.2. S prvim parametrom določimo kljuko oziroma mesto ujetja, z drugim parametrom pa določimo naziv izvajane funkcije. S tretjim parametrom določimo prioriteto izvajanja in s četrtim število argumentov, ki jih funkcija prejme.

```
function echo_comment_id($comment_id) {
   echo 'Comment ID ' . $comment_id . ' could not be found';
}
add_action( 'comment_id_not_found', 'echo_comment_id', 10, 1 );
```

Koda 2.2: Primer ujetja funkcije

Filtri so funkcije, ki se izvedejo pred izvedbo dogodka, kot je zapis podatkov v bazo, izpis obvestila na zaslon brskalnika ali pošiljanje elektronskega sporočila [16]. Filter je nameščen med podatkovno bazo in brskalnik (ko WordPress gradi strani), oziroma med brskalnik in podatkovno bazo (ko WordPress dodaja nove prispevke in komentarje v bazo). Funkcija prejme kot parameter neurejen podatek, in vrne urejenega. Večina vhodnih in izhodnih podatkov gre skozi vsaj en filter. Ravno tako vtičnik uporablja filtre za obdelavo podatkov. Tudi pri njih je potrebno paziti na poimenovanje, da ne pride do konfliktov. Filter narejen po meri se registrira z metodo `add_filter()`, kot prikazuje koda 2.3. S prvim parametrom določimo funkcijo nad katero se izvede filter, z drugim parametrom pa izvedeno funkcijo. S tretjim parametrom določimo prioriteto izvajanja in s četrtim število argumentov.

```
function display_copyright_feed($post_content) {
   if(!$copyright_text = get_option('copyright_notices_text') || !is_feed())
      return $post_content;
   return $post_content . $copyright_text;
}
add_filter('the_content','display_copyright_feed', 10, 1);
```

Koda 2.3: Primer uporabe filtra

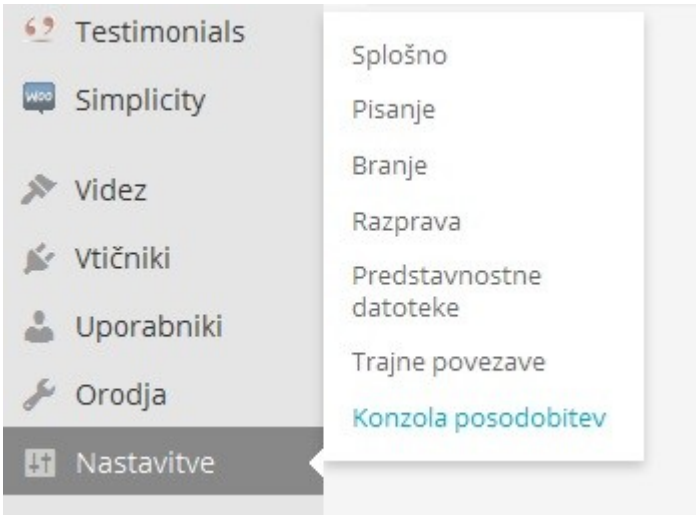
Pri dodajanju vtičnika v meni administracije je potrebno ustvariti funkcijo, ki vsebuje metodo `add_options_page()` [16], kar prikazuje koda 2.4. Metoda zgradi stran podmenija z določenim nazivom menijske vrstice in pripadajočim vtičnikom. Po potrebi se nato lahko registrirajo tudi nastavitve strani. Z akcijo `admin_menu` se menijska vrstica `Konzola posodobitev` doda v podmeni sklopa `Nastavitve`, preko katerih je dostopno delovno področje vtičnika, kot prikazuje slika 2.2. V meniju se menijska vrstica izpiše v izbranem jeziku platforme. Če jezik v vtičniku ni opredeljen oziroma preveden, se uporabi privzeti jezik vtičnika. V našem primeru angleški jezik. Vtičnik je možno dodati tudi v druge sklope menija najvišjega nivoja. Sklope in pripadajoče metode prikazuje tabela 2.1.

Naziv sklopa	**Metoda**
Nadzorna plošča (angl. *dashboard*)	`add_dashboard_page()`
Prispevki (angl. *posts*)	`add_posts_page()`
Medijske datoteke (angl. *media*)	`add_media_page()`
Strani (angl. *pages*)	`add_pages_page()`
Komentarji (angl. *comments*)	`add_comments_page()`
Videz (angl. *appearance*)	`add_theme_page()`
Vtičniki (angl. *plugins*)	`add_plugins_page()`
Uporabniki (angl. *users*)	`add_users_page()`
Orodja (angl. *tools*)	`add_management_page()`
Nastavitve (angl. *settings*)	`add_options_page()`

Tabela 2.1: Metode za dodajanje menijske opcije v sklope najvišjega nivoja menija

```php
function admin_menu_msau() {
   add_options_page(
      __( 'Multi Site Automatic Updater', 'msau' ),
      __( 'Update Console', 'msau' ),
      'manage_options',
      'multi-site-automatic-updater.php',
      'msau_options'
   );
   add_action( 'admin_init', 'register_msau_settings' );
}
add_action('admin_menu', 'admin_menu_msau');
```

Koda 2.4: Prikaz dodajanja strani vtičnika v meni in registracija nastavitev



Slika 2.2: Prikaz dostopa vtičnika v meniju

WordPress podpira večjezično predstavnost (angl. *internationalization*) [17]. Zato je smiselna vpeljava večjezičnosti tudi v vtičnik, da lahko sledi jeziku WordPressa. Jezik vtičnika se nahaja v PO in MO datotekah. To sta tekstovni datoteki, ki vsebujeta prevod teksta za izbran jezik kot prikazuje koda 2.5. Vsak jezik ima svoj par datotek. Med seboj se ločijo z oznako države oziroma oznako jezika. Za uporabo slovenskega jezika se uporablja kratica sl-

SI. Večjezično predstavnost v vtičnik vpeljemo z metodo `load_plugin_textdomain()`. S parametrom se določi identifikator in pot jezikovne datoteke. Identifikator je enolični naziv, na katerega se sklicujemo pri prejemanju prevedenih nizov. Ob zagonu vtičnika se z akcijo `init` registrira jezikovna predloga vtičnika.

```
#: multi-site-automatic-updater.php:98
#@ msau
msgid "You do not have sufficient permissions to access this page."
msgstr "Za dostop do strani potrebujete administratorski dostop."
```

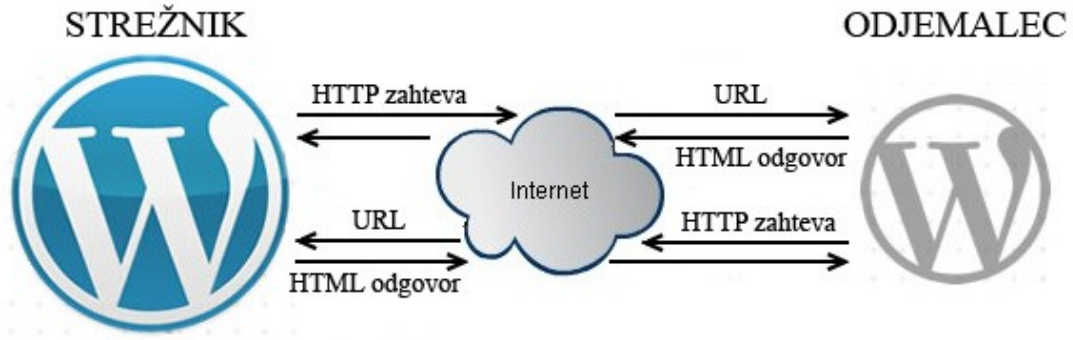
Koda 2.5: Prikaz prevoda teksta v PO datoteki

Slog CSS omogoča oblikovno prilagoditev strani. Potrebno ga je registrirati. To se stori z metodo `wp_register_style()` [17]. S parametrom se določi naziv slogovne predloge in pot do slogovne predloge. Registriran slog CSS se z metodo `wp_enqueue_style()` doda v vtičnik. S parametrom se določi naziv slogovne predloge.

JavaScript knižnjice kot je JQuery, ki jih uporabljamo v vtičniku, se registrirajo z metodo `wp_register_script()`. Registrirane knjižnice se doda v vtičnik s pomočjo metode `wp_enqueue_script()` [17].

2.3 KOMUNIKACIJA MED ODJEMALCEM IN STREŽNIKOM

Podatki, ki jih želimo prikazati v centralni enoti, se pridobijo iz odjemalcev. Da lahko izvršimo zahtevo za informacijo, moramo najprej vedeti kakšen je URL naslov spletne strani odjemalca. URL naslov pridobimo ob namestitvi vtičnika odjemalca, v platformo odjemalca. Komunikacija med strežnikom in odjemalcem mora potekati hitro, odgovori z informacijo pa morajo biti kratki in preprosti. Pri odjemalec – strežnik komunikaciji, komunikacija poteka s pomočjo metod HTTP [18]. Shemo komunikacije prikazuje slika 2.3. S prihodom različice 2.7 in z razširitvijo v različici 2.8 so metode HTTP del jedra. Pred tem je moral razvijalec sam razviti svoje metode za zahteve s pomočjo funkcij PHP. Ker se je pojavilo več izpeljank metod, je bilo vzdrževanje različnih metod zahtevno in nepregledno.



Slika 2.3: Shema odjemalec – strežnik komunikacije

Metode HTTP so:

- `wp_remote_get()`,
- `wp_remote_post()`,
- `wp_remote_head()` in
- `wp_remote_request()`.

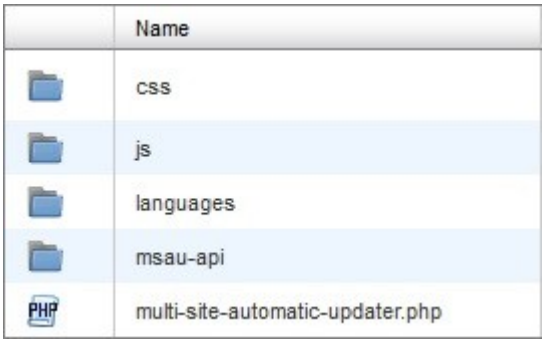
Vtičnika uporabljata metodo `wp_remote_post()`. Primer uporabe metode prikazuje koda 2.6. Zahteva se pošlje na URL naslov, določen s spremenljivko `api_url`. Dodatne parametre zahteve se določi v obliki matrike. Metoda je časovno omejena s parametrom `timeout`. Privzeta vrednost parametra je 5 sekund, zato je potrebno upoštevati čas izvedbe zahteve. Po poteku časa se povezava konča in metoda vrne odgovor o izteku zahteve. Parametre, katere želimo poslati vpišemo v polje `body` v matrični obliki. Rezultat odgovora metode je prav tako matrika. Odgovor zahteve se nahaja v polju `body`, katerega lahko prilagodimo pred izpisom na zaslonsko okno. Vtičnik strežnik skrbi za komunikacijo z odjemalci. URL naslov za komunikacijo je določen v vtičniku in se ne spreminja. Vtičnik strežnik generira zahteve za vsak odjemalec in čaka na njihov odgovor. Če je zahteva uspešno izvedena, dobi zahtevane podatke, v nasprotnem primeru dobi za odgovor napako, do katere je prišlo pri izvedbi zahteve. Odgovori se prenašajo v JSON obliki. Ko strežnik prejme odgovor, ga pretvori iz JSON oblike in prikaže na zaslonskem oknu vtičnika. Vtičnik odjemalec skrbi za izvajanje zahtev, katere posreduje strežnik. Ko odjemalec prejme zahtevo, jo poskuša izvesti. Zahteve izvaja s pomočjo lokalne platforme WordPress. Dobljen odgovor platforme pretvori v JSON obliko in pošlje strežniku kot odgovor na zahtevo.

```
$response = wp_remote_post(
   $api_url,
   array (
      'timeout' => 10,
      'body' => array (
         'action' => 'activate',
         'blogURL' => $blogURL
      )
   )
);
echo $response['body'];
```

Koda 2.6: Primer uporabe metode POST

3. VTIČNIK STREŽNIK

Vtičnik strežnik se namesti na spletno stran narejeno v WordPressu namenjeno za centralni nadzor. Lahko se namesti preko administracije ali ročno na mesto namenjeno vtičnikom. Vtičnik se nahaja v mapi pod imenom `multi-site-automatic-updater-server`. Vsebino mape vtičnika prikazuje slika 3.1. Mapa vsebuje datoteko PHP, katera se zažene ob aktivaciji vtičnika z metodo `init` in vsebuje jedro vtičnika. Jedro pri svojem delu uporablja CSS, knjižnico JQuery in jezikovno datoteko. Sklopi so ločeni tako, da se nahajajo vsak v svoji podmapi. Služijo za izgradnjo grafičnega vmesnika vtičnika. Taka struktura omogoča lažje vzdrževanje vtičnika. Za komunikacijo z odjemalcem je namenjena podmapa `msau-api` in mora biti dostopna preko spleta. V mapi se nahaja datoteka `index.php`, katera sprejema zahteve HTTP. Zahtevo za registracijo prejme kot argument `action`. *Vtičnik strežnik* na tem mestu poskuša izvesti zahtevo skupaj z morebitnimi dodatnimi argumenti. Po izvršeni zahtevi vrne primeren odgovor odjemalcu.



Slika 3.1: Razporeditev datotek vtičnika strežnika

3.1 GRAFIČNI VMESNIK

Grafični vmesnik je razdeljen na sklope s pomočjo zavihkov. Pri izgradnji zavihkov je uporabljena knjižnica JQuery in pripadajoči slog CSS. Pred prikazom grafičnega vmesnika se zajamejo vsi podatki potrebni za prikaz. Skupaj z zajemom podatkov se ustvarijo posamezni zavihki. Glavni sklopi vmesnika so: *Nastavitve*, *Seznam strani* in *Posodobitve*.

V zavihku *Nastavitve* se nahaja vnosna forma za nastavitve samodejnega posodabljanja, ki vplivajo na posodobitve WordPressa centralne strani. Formo prikazuje slika 3.2. Namesto ročne aktivacije samodejnih posodobitev v datoteki `wp-config.php`, grafični vmesnik ponudi uporabniku možnost za prijaznejši način prilagajanja nastavitev.

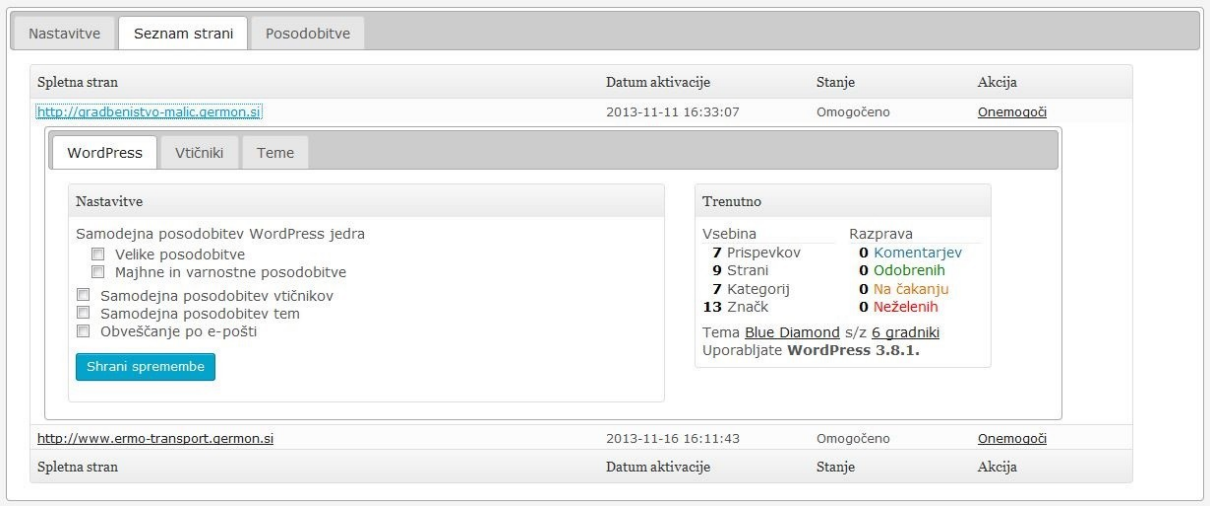
Slika 3.2: Nastavitve samodejnega posodabljanja centralne strani

Nastavitve samodejnih posodobitev so razdeljene na:

- velike posodobitve jedra,
- male posodobitve jedra,
- posodobitve vtičnikov,
- posodobitve tem,
- obvestilo o posodobitvah po elektronski pošti,
- elektronski naslov za obvestila o posodobitvah in
- vrsta poslanega obvestila.

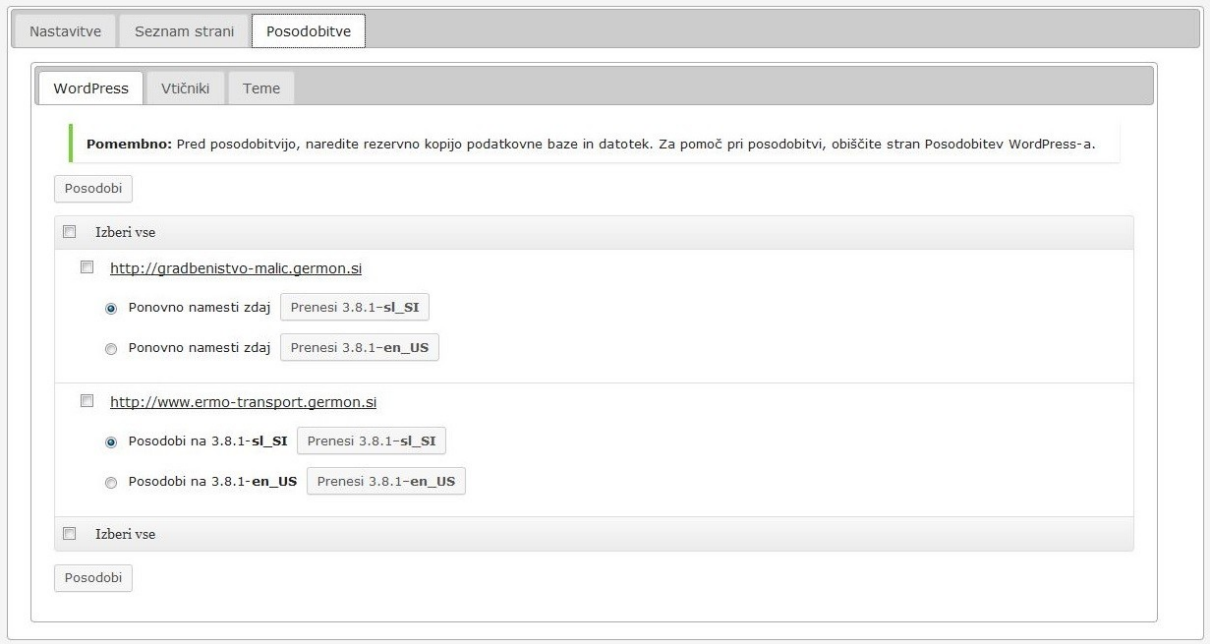
Zavihek *Seznam strani*, postreže s podatki o registraciji odjemalcev, nad katerimi se izvaja nadzor posodobitev. Podatki zajemajo URL naslov spletne strani odjemalca, datum registracije odjemalca, stanje nadzora in možnost spremembe stanja nadzora. V primeru, da je *vtičnik odjemalec* nameščen, je nadzor možno vklopiti ali izklopiti. Z vklopom samodejne posodobitve omogočimo, z izklopom pa samodejne nastavitve onemogočimo. Če *vtičnik odjemalec* ni več nameščen, se spletna stran odjemalca na seznamu ne prikaže. S klikom na povezavo spletne strani odjemalca, se prikažejo podatki, ki so razdeljeni na zavihke: *WordPress*, *Vtičniki* in *Teme*.

Zavihki prikazujejo podatke o nastavitvah samodejnega posodabljanja skupaj z osnovnimi informacijami o WordPressu, podatki o nameščenih vtičnikih in podatki o nameščenih temah, kot prikazuje slika 3.3.



Slika 3.3: Seznam odjemalcev in njihove osnovne informacije

Zavihek *Posodobitve* prikaže posodobitve jedra, vtičnikov in tem, katere so na voljo, kot prikazuje slika 3.4. Ravno tako so podatki o posodobitvah razdeljeni na posamezne zavihke: *WordPress*, *Vtičniki* in *Teme*. V tem zavihku je posodobitve možno izbrati in izvesti.



Slika 3.4: Seznam posodobitev jedra WordPress odjemalcev

3.2 NASTAVITVE ZA SAMODEJNO POSODABLJANJE CENTRALNE STRANI

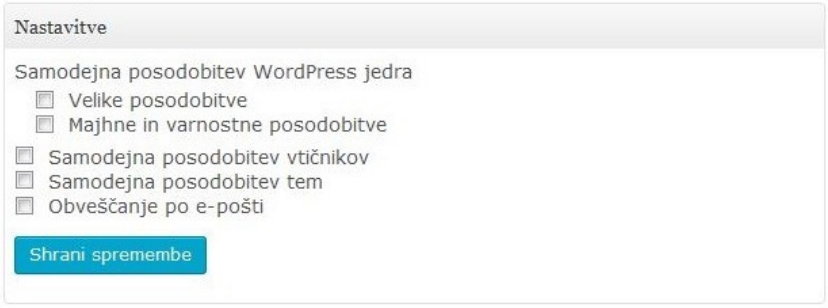
Za izvedbo samodejnih posodobitev nam WordPress nudi metode s katerimi prilagajamo izvedbo posodobitev preko filtrov.

Naredili smo funkcijo `register_msau_settings()`, v kateri smo vsako od nastavitev najprej registrirali z metodo `register_setting()`. Parameter `option_group` določa, kateri skupini možnosti nastavitev pripada, parameter

`option_name` pa naziv nastavitve. Po izgradnji strani podmenija smo z metodo `admin_init` dodali nastavitve samodejnega posodabljanja. Vrednosti nastavitev smo iz podatkovne baze pridobili z metodo `get_option` in z njimi izpolnili vnosno formo, katero prikazuje slika 3.5.

S klikom na gumb *Shrani*, forma pošlje spremenjene podatke na stran `option.php`, kjer se podatki shranijo v podatkovno bazo.

Samodejne posodobitve lahko tudi izklopimo v celoti. To storimo tako, da metodo `automatic_updater_disabled` s pomočjo filtra omogočimo oziroma jih vklopimo, če metodo s pomočjo filtra onemogočimo. Za omogočanje in onemogočanje preko filtrov nam WordPress postreže s funkcijama `__return_true()` in `__return_false()`. Za potrebe posodabljanja smo naredili funkcijo `msau_automatic_updates()`, katera opravi proces nastavljanja nastavitev in njihovo aktivacijo. Funkcija najprej metodo `automatic_updater_disabled` izklopi. Prebere izbrane nastavitve in na podlagi pridobljenih podatkov metode za posodabljanje s pomočjo filtrov vključi oz. izključi. Metode prikazuje in opisuje tabela 3.1.



Slika 3.5: Vnosna forma samodejnih nastavitev centralne strani

Metoda	**Opis metode**
`allow_major_auto_core_updates`	Velike posodobitve jedra
`allow_minor_auto_core_updates`	Male posodobitve jedra
`auto_update_plugin`	Posodobitve vtičnikov
`auto_update_theme`	Posodobitve tem
`auto_core_update_send_email`	Obvestilo o posodobitvah po elektronski pošti
`auto_core_update_email`	Elektronski naslov za obvestila o posodobitvah
`automatic_updates_debug_email`	Vrsta poslanega obvestila

Tabela 3.1: Metode za nadzor samodejnih posodobitev preko filtrov

Privzeti elektronski naslov je elektronski naslov administratorja. Ker želimo, da obvestilo o posodobitvah prejme drug prejemnik, smo dodali metodo za prepis elektronskega naslova `override_update_email()`, ki upošteva elektronski naslov iz nastavitev, kot prikazuje koda 3.1. Preverjanje novih posodobitev se zgodi vsakih 12 ur. Če obstaja nova posodobitev za jedro, vtičnik ali temo in dovolimo posodobitev, se ta samodejno izvede. Po končanem

posodabljanju se na elektronski naslov, ki smo ga navedli pošlje obvestilo o uspešnosti izvedenih posodobitev.

```
function override_update_email($email) {
   $email['to'] = get_option('notificationUpdateEmailAddress');
   return $email;
}
add_filter( 'auto_core_update_email','override_update_email',1,1);
```

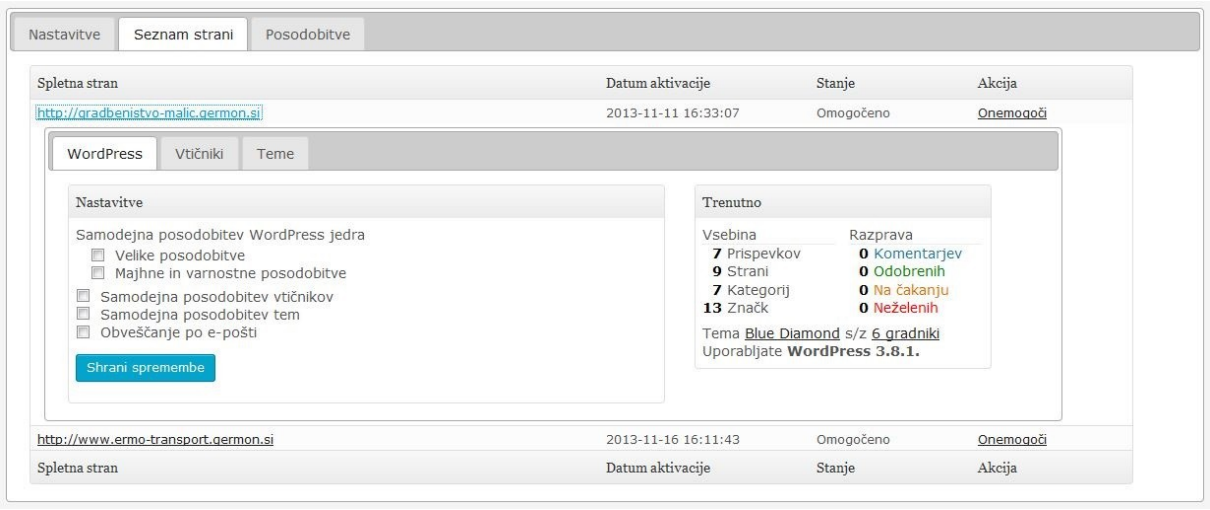
Koda 3.1: Primer zamenjave elektronskega naslova z uporabo filtra

3.3 REGISTRACIJA ODJEMALCA

Pri aktivaciji *vtičnika strežnika* se v podatkovni bazi MySQL ustvari tabela `wp_wp_site` z uporabo funkcije `dbDelta()` [18]. V tabelo se zapisujejo podatki o registraciji odjemalcev. Ko *vtičnik strežnik* dobi zahtevo po registraciji odjemalca, mu ta sporoči svoj URL naslov. URL naslov se zapiše v podatkovno bazo. Če naslov že obstaja, potem se spremeni le stanje v aktivno. *Vtičnik strežnik* vrne odgovor odjemalcu o uspešnosti registracije odjemalca.

3.4 PRIKAZ PODATKOV O WORDPRESSU ODJEMALCEV

Zavihek WordPress nam postreže s formo o nastavitvah samodejnega posodabljanja odjemalca in z osnovnimi podatki WordPressa odjemalca. Zavihek prikazuje slika 3.6. Ne da bi bilo potrebno vstopanje v administracijo odjemalca, je iz tega mesta možna sprememba nastavitev za samodejno posodabljanje in hiter pregled stanja WordPressa odjemalca.



Slika 3.6: Nastavitve samodejnih posodobitev in osnovne informacije WordPressa

Pred prikazom izbranih nastavitev *vtičnik strežnik* preveri stanje nastavitev za samodejno posodabljanje pri odjemalcu z zahtevo po zajemu podatkov o samodejnih nastavitvah. Odgovor s trenutnimi nastavitvami v JSON obliki najprej dekodira in nato s podatki izpolni vnosna polja nastavitev. Za shranjevanje sprememb *vtičnik strežnik* pošlje zahtevo z novimi

podatki odjemalcu. *Vtičnik odjemalec* pošlje odgovor o uspešnosti shranjevanja podatkov. Odgovor se prikaže na vrhu prikazanega okna.

Podatki trenutnega stanja WordPressa so razdeljeni na vsebino, razpravo, temo z uporabljenimi gradniki in različico jedra WordPress. Podatki vsebine so:

- število prispevkov,
- število strani,
- število kategorij in
- število značk.

Podatki vsebine prikažejo strukturo spletne strani. S hitrim pregledom vsebine je možno ugotavljanje stanja in spremembe spletne strani. Pomaga pri določitvi zahtevnosti spletne strani. Podatki razprave so:

- število vseh komentarjev,
- število odobrenih komentarjev,
- število komentarjev na čakanju in
- število neželenih komentarjev.

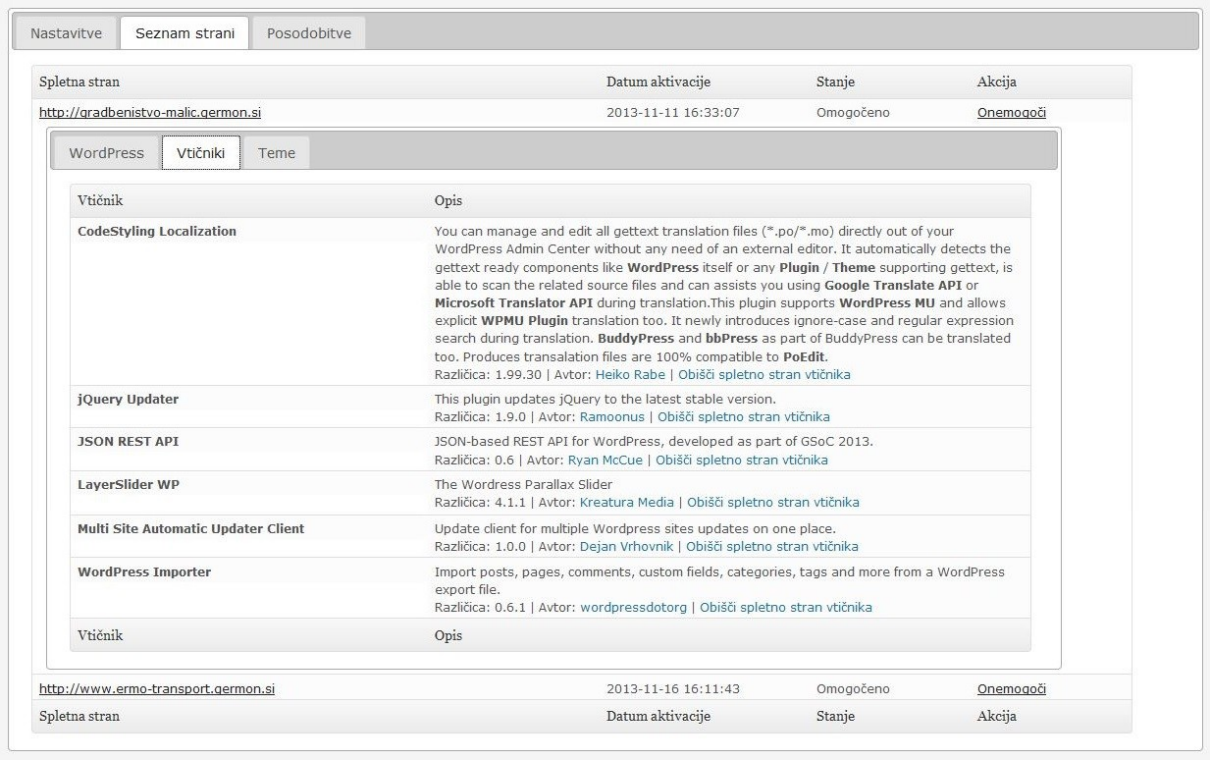
Podatki o razpravi nudijo hitro informacijo o komentarjih na spletni strani. Povejo nam ali spletna stran potrebuje naše posredovanje pri urejanju komentarjev.

Strnjen prikaz informacije o različici WordPressa ter podatki o nameščeni temi, njeni različici in njeni strukturi, služijo hitremu preverjanju stanja jedra in teme. Prikaže se aktivna tema strani. Za prikaz posameznega sklopa podatkov je potrebna posamezna zahteva odjemalcu in s tem posamezen odgovor. Prednost manjših zahtev je prilagodljivost in optimalen zajem podatkov.

3.5 PRIKAZ PODATKOV O NAMEŠČENIH VTIČNIKIH ODJEMALCEV

V zavihku *Vtičniki* je seznam nameščenih vtičnikov odjemalca, kar prikazuje slika 3.7. Za informacijo o nameščenih vtičnikih pošlje vtičnik strežnik zahtevo odjemalcu. Odjemalec odgovori s seznamom vtičnikov. S pridobljenimi podatki nato ustvarimo grafični seznam vtičnikov. Seznam prikazuje:

- naziv vtičnika,
- opis vtičnika,
- različico vtičnika,
- naziv in spletno stran avtorja in
- spletno stran vtičnika.

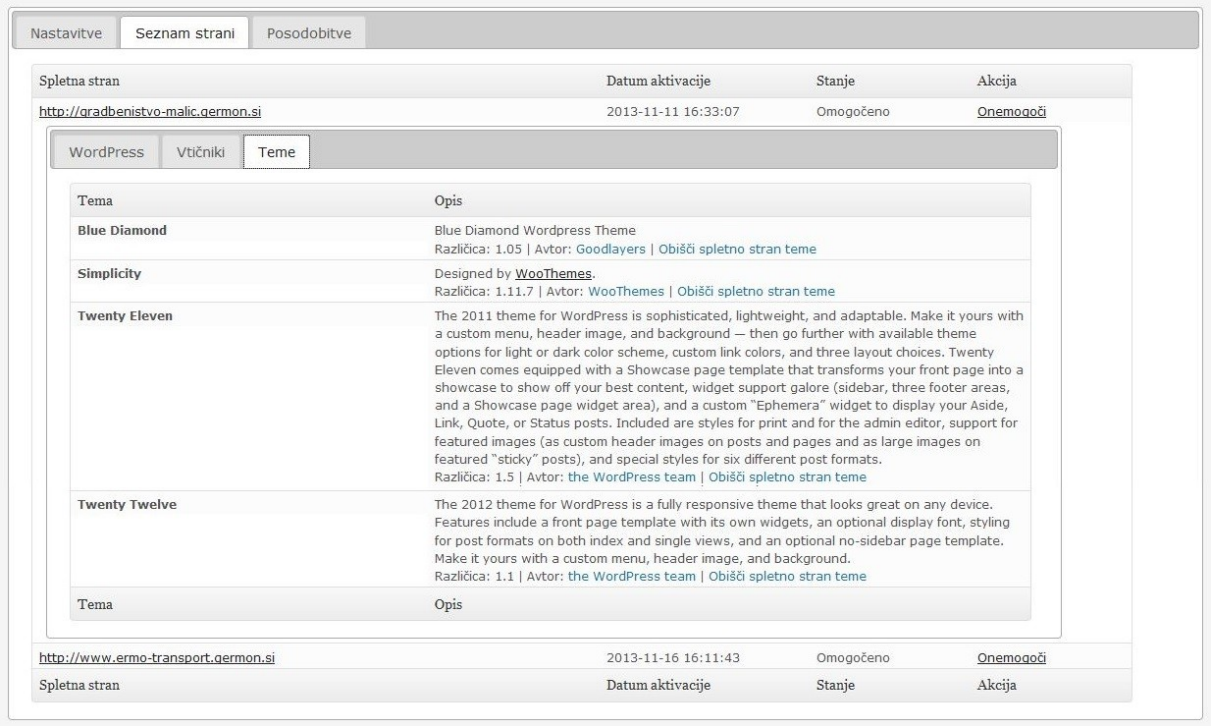


Slika 3.7: Seznam nameščenih vtičnikov odjemalca

3.6 PRIKAZ PODATKOV O NAMEŠČENIH TEMAH ODJEMALCEV

V zavihku *Teme* je seznam nameščenih tem odjemalca. Zavihek prikazuje slika 3.8. Za pridobitev informacije o nameščenih temah pošlje *vtičnik strežnik* zahtevo po podatkih *vtičniku odjemalcu*. Odjemalec odgovori s seznamom nameščenih tem. Pridobljene podatke prikažemo grafično. Seznam prikazuje:

- naziv teme,
- opis teme,
- različico teme,
- naziv in spletno stran avtorja ter
- spletno stran teme.



Slika 3.8: Seznam nameščenih tem odjemalca

3.7 SEZNAM ROČNIH POSODOBITEV JEDER ODJEMALCEV

V zavihku *WordPress* prikažemo za odjemalce, kateri so pod nadzorom posodobitve jedra WordPressa, če so na voljo, kar prikazuje slika 3.9. Seznam odjemalcev pridobimo iz podatkovne baze. Za vsakega odjemalca prikažemo možno posodobitev različice jedra z jezikovno podporo in različico jedra brez jezikovne podpore, se pravi angleško verzijo različice. Pri vsaki opciji je povezava, s katere lahko prenesemo posodobitev na računalnik.

Vsakemu odjemalcu se pošlje zahteva naj preveri, če obstaja posodobitev jedra. *Vtičnik odjemalec* odgovori s stanjem posodobitev jedra. Če posodobitev ne obstaja, prikažemo možnost ponovne posodobitve obstoječe različice jedra z jezikovno podporo in angleško različico ponovne posodobitve.

Seznam nam omogoča hiter izbor vseh odjemalcev. Za posamezno posodobitev jedra je potrebna izbira odjemalca in različico posodobitve. Privzeta je različica z jezikovno podporo.

Ko uporabnik označi posodobitve jeder, se ob kliku na gumb *Posodobi*, pošlje zahteva posameznemu *vtičniku odjemalcu* naj izvede posodobitev. Odgovor o uspešnosti posodobitve za vsakega odjemalca prikažemo kot obvestilo na vrhu zaslonskega okna. Pri pošiljanju zahteve je potrebno paziti na nastavljen čas trajanja zahteve, saj posodobitev potrebuje nekaj več časa za izvedbo. Čas izvedbe posamezne zahteve je nastavljen na 10 minut.

Slika 3.9: Seznam ročnih posodobitev jeder odjemalcev

3.8 SEZNAM ROČNIH POSODOBITEV VTIČNIKOV ODJEMALCEV

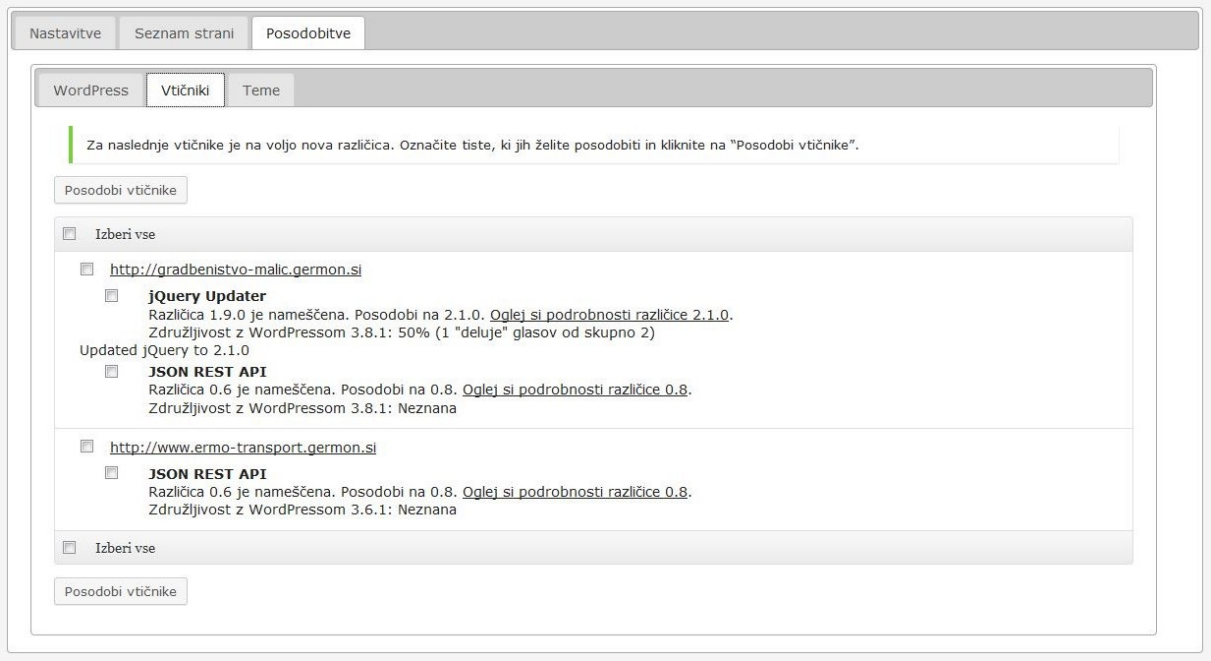
V zavihku *Vtičniki* prikažemo za odjemalce iz podatkovne baze seznam posodobitev vtičnikov, kot prikazuje slika 3.10. Vsakemu odjemalcu iz seznama se pošlje zahteva naj preveri, če so na voljo posodobitve za nameščene vtičnike. Odjemalec odgovori s posodobitvami vtičnikov. Posodobitve prikažemo na seznamu posodobitev.

Prikazani podatki o posodobitvi vtičnika:

- naziv vtičnika,
- nameščena različica vtičnika,
- različica posodobitve vtičnika,
- podrobnosti posodobitve in
- združljivost vtičnika z WordPressom.

Pri izbiri lahko označimo posamezno posodobitev, katero želimo izvesti. Seznam omogoča tudi hitro izbiro vseh posodobitev vtičnikov vseh odjemalcev in izbiro vseh vtičnikov posameznega odjemalca.

Ko uporabnik označi posodobitve vtičnikov, se ob kliku na gumb *Posodobi vtičnike* pošlje vsakemu odjemalcu zahteve, naj izvede posodobitve. Za posamezen vtičnik se pošlje posamezna zahteva. Odjemalec pošlje odgovor o uspešnosti izvedene posodobitve. Odgovori se izpišejo kot obvestila na vrhu zaslonskega okna.



Slika 3.10: Seznam ročnih posodobitev vtičnikov odjemalcev

3.9 SEZNAM ROČNIH POSODOBITEV TEM ODJEMALCEV

Na zavihku *Teme* prikažemo za odjemalce iz podatkovne baze seznam posodobitev tem, kot prikazuje slika 3.11. Vsakemu odjemalcu iz seznama se pošlje zahteva, naj preveri, če so na voljo posodobitve za nameščene teme. Odjemalec vrne informacijo o posodobitvah tem. Pridobljene podatke izpišemo na seznamu posodobitev tem za vsakega odjemalca.

Prikazani podatki o posodobitvi vtičnika:

- naziv teme,
- različica nameščene teme in
- različica posodobitve teme.

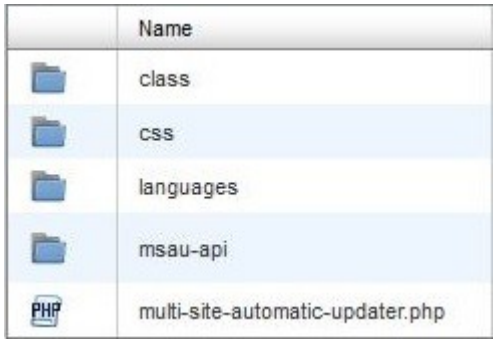
Na seznamu posodobitev tem označimo posamezno posodobitev za temo, katero želimo izvesti, izberemo vse posodobitve tem za vse odjemalce oziroma izberemo vse posodobitve posameznega odjemalca.

Ob kliku na gumb *Posodobi teme* se pošlje vsakemu odjemalcu zahteve, naj izvede označene posodobitve tem. Za posamezno temo se pošlje posamezna zahteva. Odgovori se izpišejo kot obvestila na vrhu zaslonskega okna.

Slika 3.11: Seznam ročnih posodobitev tem odjemalcev

4. VTIČNIK ODJEMALEC

Vtičnik odjemalec se namesti na spletno stran narejeno v WordPressu, nad katero želimo izvajati nadzor posodobitev. Domača mapa vtičnika je `multi-site-automatic-updater-client`. Vsebino mape prikazuje slika 4.1. Struktura je enaka kot pri *vtičniku strežniku*. V mapi `class` se nahajajo razredi za posodobitev vtičnikov in tem. Naloga *vtičnika odjemalca* je sprejem zahtev, njihova izvedba in vračanje odgovorov. Zahteve sprejema v podmapi `msau-api`.



Slika 4.1: Razporeditev datotek vtičnika odjemalca

4.1 ZAHTEVA ZA REGISTRACIJO ODJEMALCA

Ob aktivaciji *vtičnika odjemalca* se samodejno pošlje zahteva za registracijo odjemalca *vtičniku strežniku*, kateri odgovori z uspešnostjo registracije. Zahteva vsebuje podatek o spletnem naslovu odjemalca. Informacijo dobimo z uporabo funkcije `get_bloginfo()` in argumentom `wpurl`. Odgovor o uspešnosti registracije se izpiše kot obvestilo na prikazanem oknu *vtičnika odjemalca*.

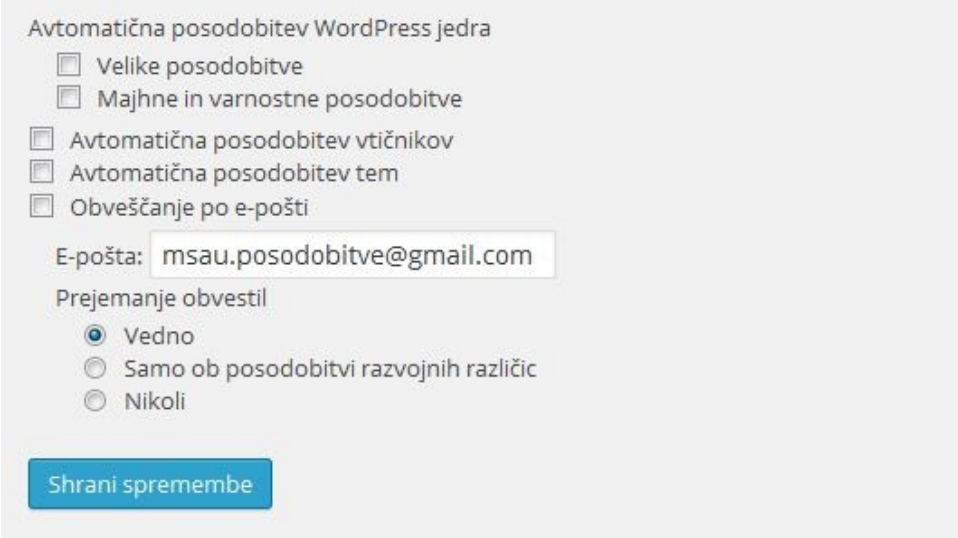
4.2 GRAFIČNI VMESNIK

Grafični vmesnik je preprosta forma za urejanje nastavitev samodejnih posodobitev, ki jo prikazuje slika 4.2.

Nastavitve samodejnih posodobitev so razdeljene na:

- velike posodobitve jedra,
- male posodobitve jedra,
- posodobitve vtičnikov,
- posodobitve tem,
- obvestilo o posodobitvah po elektronski pošti,
- elektronski naslov za obvestila o posodobitvah in
- vrsta poslanega obvestila.

Podoba in delovanje forme sta enaki kot pri *vtičniku strežniku*.



Slika 4.2: Nastavitve samodejnih posodobitev odjemalca

4.3 NASTAVITVE ZA SAMODEJNO POSODABLJANJE ODJEMALCA

Upravljanje z nastavitvami pri *vtičniku odjemalcu* je možno na dva načina: lokalno ali iz oddaljenega mesta.

Delovanje mehanizma je v obeh primerih enako, kot pri *vtičniku strežniku*. Razlika je v tem, da se podatki za nastavitve pri lokalnem načinu preberejo iz forme, za oddaljeno mesto pa iz zahteve, katero je poslal *vtičnik strežnik*. Odgovor se v prvem primeru prikaže kot obvestilo na prikazanem zaslonu *vtičnika odjemalca*, v drugem primeru se odgovor pošlje *vtičniku strežniku*.

4.4 ZAJEM PODATKOV O NAMEŠČENEM WORDPRESSU ODJEMALCA

Zajem podatkov se zgodi v več korakih z različnimi zahtevami:

- zajem nastavitev za samodejno posodobitev iz oddaljenega mesta,
- shranjevanje sprememb nastavitev za samodejno posodobitev,
- zajem stanja vsebine WordPressa,
- zajem stanja razprave in
- zajem aktivirane teme z gradniki in različico WordPressa.

Zajem podatkov bi lahko zajeli samo z eno zahtevo. S tem bi odgovor zahteve postal bolj zahteven, vendar bi v primeru napake zahteve izgubili tudi večjo količino podatkov, potrebnih za prikaz na centralni stani. Z razčlenitvijo velike kompleksne zahteve na več manjših enostavnih zahtev, smo tveganje izgube podatkov ob napaki zahteve omejili na posamezno zahtevo. Hkrati pa je preprosto zahtevo lažje ponovno uporabiti v drugih primerih prikaza.

Na zahtevo zajema nastavitev za samodejno posodobitev iz oddaljenega dostopa, se z metodo `get_option()` preberejo vrednosti nastavitev iz podatkovne baze. Primer uporabe

metode prikazuje koda 4.1 Podatke je potrebno najprej pretvoriti v matrični zapis in nato zakodirati v JSON obliko. Podatke se pošlje kot odgovor na zahtevo *vtičniku strežniku*.

```
get_option('majorCoreUpdate');
```

Koda 4.1: Primer zajetja vrednosti nastavitve iz podatkovne baze

Na zahtevo shranjevanja sprememb nastavitev za samodejno posodobitev se z metodo `update_option()` posodobi vrednosti nastavitev v bazi in z metodami za nadzor samodejnih posodobitev preko filtrov prilagodi samodejno posodabljanje. Postopek shranjevanja nastavitve prikazuje koda 4.2. *Vtičniku strežniku* se vrne odgovor o uspešnosti izvedenih sprememb.

```
update_option('majorCoreUpdate', $_POST['majorCoreUpdate']);
if ($_POST['majorCoreUpdate'] == 1) {
   add_filter('allow_major_auto_core_updates', '__return_true', 1);
} else {
   add_filter('allow_major_auto_core_updates', '__return_false', 1);
}
```

Koda 4.2: Postopek shranjevanja nastavitve samodejnega posodabljanja

Na zahtevo zajema stanja vsebine zajamemo podatke stanja WordPressa. Metoda `wp_count_posts()` pridobi število prispevkov in število strani. S poslanim argumentom `post` za število prispevkov in `page` za število strani. Z metodo `wp_count_terms()` pridobimo število kategorij in število značk. Z argumentom `category` pridobimo število kategorij in z argumentom `post_tag` število značk. Pridobljene podatke pretvorimo v JSON obliko in pošljemo kot odgovor na zahtevo *vtičniku strežniku*.

Na zahtevo zajema stanja razprave, zajamemo podatke o komentarjih s pomočjo metode `wp_count_comments()`. Metoda vrne matriko komentarjev glede na stanje. Odgovor pošljemo *vtičniku strežniku* v JSON obliki.

Na zahtevo zajema aktivne teme, gradnikov in različice WordPressa se zajame informacijo o nazivu teme. Metoda `wp_get_theme()` vrne matriko podatkov o temi. Da izluščimo naziv teme uporabimo še funkcijo `get()` z argumentom `Name`. Gradnike dobimo z metodo `wp_get_sidebars_widgets()`, jih preštejemo, saj nas zanima število nameščenih gradnikov. Metoda `get_bloginfo()` z argumentom `version` nam vrne nameščeno različico jedra WordPressa. Zbrane podatke pretvorimo v JSON obliko in pošljemo *vtičniku strežniku* kot odgovor.

4.5 ZAJEM PODATKOV O NAMEŠČENIH VTIČNIKIH ODJEMALCA

Na zahtevo zajema nameščenih vtičnikov uporabimo metodo `get_plugins()`. Metoda vrne seznam nameščenih vtičnikov s podrobnimi informacijami o vtičniku. Matriko, katero vrne metoda pretvorimo v JSON obliko in jo posredujemo *vtičniku strežniku* kot odgovor.

4.6 ZAJEM PODATKOV O NAMEŠČENIH TEMAH ODJEMALCA

Na zahtevo zajema podatkov o nameščenih temah zajamemo seznam nameščenih tem z metodo `wp_get_themes()`. Podrobne informacije o temi pa z metodo `get_theme_data()`. Tako dobimo seznam nameščenih tem s podrobnostmi. Podatke pretvorimo v JSON obliko in posredujemo *vtičniku strežniku* kot odgovor.

4.7 ROČNA POSODOBITEV JEDRA ODJEMALCA

Na zahtevo zajema informacije o posodobitvi jedra, se izvede metoda `wp_version_check()`. Metoda preveri, če obstajajo posodobitve za nameščeno različico jedra WordPressa. Z metodo `get_core_update()` dobimo seznam posodobitev, ki so na voljo. Zgradimo matriko podatkov, ki vsebujejo podatke trenutno nameščene različice jedra in podatke nove različice jedra. Matriko pošljemo v JSON obliki *vtičniku strežniku*.

Na zahtevo posodobitve jedra, se izvede metoda `do_core_upgrade()`. Ker metoda posega v datotečni sistem, moramo imeti pravice za dostopanje in urejanje le tega. To dosežemo s prijavo uporabnika administracije WordPressa z metodo `wp_login`. Uprabnik administracije mora imeti pravice za posodobitev jedra. V zahtevi se nahaja informacija s katero različico želimo posodobiti jedro WordPressa. Rezultat o uspešnosti posodobitve jedra pošljemo kot odgovor *vtičniku strežniku*.

4.8 ROČNA POSODOBITEV VTIČNIKOV ODJEMALCA

Na zahtevo zajema posodobitev vtičnikov, se izvede metoda `get_plugins()`. Metoda vrne seznam nameščenih vtičnikov. Z metodo `get_site_transistent()` in argumentom `update_plugins` preverimo, če obstajajo posodobitve za nameščene vtičnike. Seznam posodobitev vtičnikov se v JSON obliki pošlje *vtičniku strežniku* kot odgovor.

Na zahtevo zajema podrobnosti posodobitve vtičnika, se izvede metoda `plugins_api()` z argumentoma `plugin_information` in `slug`. Metoda vrne podrobnosti posodobitve vtičnika, katero vrnemo kot odgovor v JSON obliki.

Na zahtevo izvedbe posodobitev vtičnikov, se izvede metoda `bulk_upgrade()` s parametrom `plugin`. Uporabo metode prikazuje koda 4.3. Parameter pove, kateri vtičnik želimo posodobiti. Metoda se nahaja v razredu `Plugin_Upgrader`. Ker metoda dostopa v datotečni sistem, je potrebna prijava uporabnika z metodo `wp_login`. Uporabnik mora imeti pravice za posodabljanje vtičnikov. Metoda vrne obvestilo o uspešnosti posodobitve vtičnika, katerega pošljemo kot odgovor *vtičniku strežniku*. Po posodobitvi preverimo z metodo `wp_update_plugins()`, če je še kakšna posodobitev vtičnika. Zahteva se pošlje in izvede za vsak vtičnik posebej.

```
$upgrader = new Plugin_Upgrader(
   new Bulk_Plugin_Upgrader_Skin(compact('nonce', 'url'))
);
$upgrader->bulk_upgrade((array)$plugin);
```

Koda 4.3: Prikaz uporabe metode za posodobitev vtičnikov

4.9 ROČNA POSODOBITEV TEM ODJEMALCA

Na zahtevo zajema posodobitev tem, se izvede metoda `get_theme_updates()`. Metoda vrne seznam posodobitev tem, katerega vrnemo kot odgovor v JSON obliki *vtičniku strežniku*.

Na zahtevo zajema podrobnosti teme, se izvede metoda `get_theme_data()` kot parameter prejme pot do sloga CSS teme. Rezultat metode so podrobne informacije o temi in jih kot odgovor pošljemo v JSON obliki *vtičniku strežniku*.

Na zahtevo izvedbe posodobitev tem, se izvede metoda `bulk_upgrade()` s parametrom `theme`. Uporabo metode prikazuje koda 4.4. Parameter pove, katero temo moramo posodobiti. Metoda se nahaja v razredu `Theme_Upgrader`. Ker metoda dostopa v datotečni sistem, je potrebna prijava uporabnika z metodo `wp_login`. Uporabnik mora imeti pravice za posodabljanje tem. Metoda vrne obvestilo o uspešnosti posodobitve teme, katerega pošljemo kot odgovor *vtičniku strežniku*. Po posodobitvi preverimo z metodo `wp_update_themes()`, če je še kakšna posodobitev teme. Zahteva se pošlje in izvede za vsako temo posebej.

```
$upgrader = new Theme_Upgrader(
   new Bulk_Theme_Upgrader_Skin(compact('nonce', 'url'))
);
$upgrader->bulk_upgrade((array)$theme);
```

Koda 4.4: Prikaz uporabe metode za posodobitev tem

5. SKLEP

V okviru diplomske naloge smo izdelali rešitev za centralni nadzor posodobitev platforme WordPress, ki omogoča samodejno in ročno posodobitev jedra, vtičnikov ter tem za večje število med seboj neodvisnih spletnih strani. Rešitev omogoča prikaz trenutnega stanja nameščenega jedra, vtičnikov ter tem. Z izdelavo rešitve smo želeli poenotiti mesto izvajanja posodobitev in s tem skrajšati čas pregleda in izvedbo posodobitev.

Na začetku razvoja smo morali raziskati delovanje platforme in medsebojno komunikacijo. Uporabili smo metode, ki jih platforma že uporablja za prikaz stanja jedra, vtičnikov in tem za posodobitev posamezne platforme. Za izvedbo ročnih posodobitev smo morali nekatere metode prilagoditi centralnemu sistemu. Paziti smo morali, da s prilagoditvami ne okrnemo neodvisnosti posameznega sistema. Končna rešitev se je izkazala za enostavnejšo in hitrejšo kot v primeru posameznega posodabljanja strani, omogočala pa je tudi izvajanje posodobitev pri večjemu številu spletnih strani.

Pri ročnem posodabljanju se postopek prične z vstopom v posamezno administracijo platforme, nadaljuje s preverjanjem novih posodobitev in konča z izvedbo izbranih posodobitev v paketu. Kot prikazuje tabela 5.1 smo analizirali časovno zahtevnost izvedbe posodobitev, kjer smo zajeli različno število spletnih strani. Za vstop v administracijo, preverjanje in izbor novih posodobitev smo porabili 30 sekund, izvedba posodobitev pa je trajala 15 sekund. Čas vstopanja v administracijo se je podaljševal z vsakim novim vstopom (tudi za 100%), saj je iskanje pravega uporabniškega imena in gesla ter vnos le-tega za uporabnika zelo zamudno in zahteva veliko njegove koncentracije. Z vsakim vnosom se povečuje tudi verjetnost uporabniške napake, kar še podaljša postopek posodobitve.

Pri ročni izvedbi posodobitev s centralnim nadzorom se postopek prične z vstopom v administracijo centralne strani, nadaljuje se z zajemom novih posodobitev in izvedbo izbranih posodobitev v paketu. Čas izvedbe posodobitev je ostal enak, izničili pa smo čas vstopa v administracijo. S tem smo bistveno pohitrili izvedbo posodobitev kar je razvidno iz tabele 5.1. Odstranili smo tudi možnost uporabniške napake pri vstopu v administracijo. Uporabniku smo z rešitvijo omogočili lažjo in hitrejšo izvedbo posodobitev. Ozko grlo rešitve je začetni zajem podatkov, ki je odvisen od števila zajetih spletnih strani. Z optimizacijo zajema podatkov je možna še dodatna pohitritev rešitve.

Število spletnih strani	Časovna zahtevnost ročne izvedbe posodobitev	Časovna zahtevnost izvedbe ročnih posodobitev s centralnim nadzorom
1	45s	45s
10	520s	160s
50	3390s	780s

Tabela 5.1: Primerjava časovne zahtevnosti izvedbe posodobitev s centralnim nadzorom

Z vklopom samodejnih posodobitev je postopek popolnoma avtomatiziran. Potrebno je le redno preverjanje elektronskih sporočil in pravilno delovanje izvedenih posodobitev.

Zastavljena implementacija rešitve ponuja še obilo možnosti za nadgradnjo in izboljšave, predvsem zaradi nenehne nadgradnje jedra in novih funkcionalnosti platforme.

Izdelano rešitev bi lahko razširili in nadgradili na več področjih.

Za izvedene posodobitve bi lahko hranili zgodovino posodobitev. Obvestilo, ki se prikaže ob izvedbi posodobitve jedra, vtičnikov in tem, bi shranili še v podatkovni bazi in tako omogočili kronološki pregled posodobitev.

Namesto zavihkov bi lahko ustvarili strani in tako optimizirali zajem podatkov. Zajeli bi samo podatke za prikaz izbrane strani.

Komunikacija med centralno stranjo in nadzorovano stranjo bi lahko dodatno zaščitili, kar pripomore k varnejši uporabi prenosa podatkov.

Podobno rešitev odjemalca bi bilo možno implementirati tudi za druge sisteme CMS (recimo Joomlo).

LITERATURA

[1] (2014) WordPress.org Slovenija. Dostopno na: http://sl.wordpress.org/.

[2] (2014) Joomla! The CMS Trusted By Millions for their Websites. Dostopno na: http://www.joomla.org/.

[3] (2014) TYPO3 - The Enterprise Open Source CMS - TYPO3 - The Enterprise Open Source CMS. Dostopno na: http://typo3.org/.

[4] (2014) Drupal - Open Source CMS Drupal.org. Dostopno na: https://drupal.org/.

[5] (2014) PrestaShop: Ecommerce Software to create your Online Store. Dostopno na: http://www.prestashop.com/en/home/.

[6] (2014) b2 - a classy weblog tool. Dostopno na: http://cafelog.com/.

[7] (2014) Easily Monitor and Manage all of your WordPress Sites with WP Remote. Dostopno na: https://wpremote.com/.

[8] (2014) InfiniteWP - Manage Multiple WordPress Sites. Dostopno na: https://infinitewp.com/.

[9] (2014) Manage WordPress Sites from One Dashboard - ManageWP.com. Dostopno na: https://managewp.com/.

[10] (2014) iControlWP - Manage Multiple WordPress Sites With Ease. Dostopno na: http://www.icontrolwp.com/.

[11] (2014) Bulk Manage WordPress - Manage Multiple WordPress Blogs From One Dashboard. Dostopno na: http://cmscommander.com/.

[12] (2014) Notepad++ Home. Dostopno na: http://notepad-plus-plus.org/.

[13] (2014) Tabs | jQuery UI. Dostopno na: http://jqueryui.com/tabs/.

[14] (2014) WordPress > Slovenija << Izidi. Dostopno na: http://sl.wordpress.org/releases/#older/.

[15] (2014) Main Page << WordPress Codex. Dostopno na: http://codex.wordpress.org/.

[16] B. Williams, D. Damstra in H. Stern, Professionl WordPress: Design and Developement,

2nd edition, Wrox, 2013.

[17] A. Brazell, WordPress Bible, Wiley, 2010.

[18] B. Williams, O. Richard in J. Tadlock, Professional WordPress: Plugin Developement, Wrox, 2011.